Methods and Apparatus for Background Memory Management

by inventors Mario Nemirovsky, Naren Sankar,
Adolfo Nemirovsky and Enric Musoll

Field of the Invention

The present invention is in the area of integrated circuit microprocessors, and pertains in particular to memory management, and the use of microprocessor resources in such management.

Background of the Invention

Microprocessors, as is well-known in the art, are integrated circuit (IC) devices that are enabled to execute code sequences which may be generalized as software. In the execution most microprocessors are capable of both logic and arithmetic operations, and typically modern microprocessors have on-chip resources (functional units) for such processing.

Microprocessors in their execution of software strings typically operate on data that is stored in memory. This data needs to be brought into the memory before the processing is done, and sometimes needs to be sent out to a device that needs it after its processing.

There are in the state-of-the-art two well-known mechanisms to bring data into the memory and send it out to a device when necessary. One mechanism is loading and storing the data through a sequence of Input/Output (I/O) instructions. The other is through a direct-memory access device (DMA).

In the case of a sequence of *I/O* instructions, the processor spends significant resources in explicitly moving data in and out of the memory. In the case of a DMA system, the processor programs an external hardware circuitry to perform the data transferring. The DMA circuitry performs all of the required memory accesses to perform the data transfer to and from the memory, and sends an acknowledgement to the processor when the transfer is completed.

In both cases of memory management in the art the processor has to explicitly perform the management of the memory, that is, to decide whether the desired data structure fits into the available memory space or does not, and where in the memory to store the data. To make such decisions the processor needs to keep track of the regions of memory wherein useful data is stored, and regions that are free (available for data storage). Once that data is processed, and sent out to another device or location, the region of memory formerly associated with the data is free to be used again by new data to be brought into memory. If a data structure fits into the available memory, the processor needs to decide where the data structure will be stored. Also, depending on the requirements of the processing, the data structure can be stored either consecutively, in which case the data structure must occupy one of the empty regions of memory; or non-consecutively, wherein the data structure may be partitioned into pieces, and the pieces are then stored into two or more empty regions of memory.

An advantage of consecutively storing a data structure into memory is that the accessing of this data becomes easier, since only a pointer to the beginning of the data is needed to access all the data.

When data is not consecutively stored into the memory, access to the data becomes more difficult because the processor needs to determine the explicit locations of the specific bytes it needs. This can be done either in

3

software (i.e. the processor will spend its resources to do this task) or in hardware (using a special circuitry). A drawback of consecutively storing the data into memory is that memory fragmentation occurs. Memory fragmentation happens when the available chunks of memory are smaller than the data structure that needs to be stored, but the addition of the space of the available chunks is larger than the space needed by the data structure. Thus, even though enough space exists in the memory to store the data structure, it cannot be consecutively stored. This drawback does not exist if the data structure is allowed to be non-consecutively stored.

Still, a smart mechanism is needed to generate the lowest number of small regions, since the larger the number of small regions that are used by a data structure, the more complex the access to the data becomes (more specific regions need to be tracked) regardless of whether the access is managed in software or hardware as explained above.

What is clearly needed is system for background management of memory in systems where large amounts of data must be moved to and from a memory for processing.

Summary of the Invention

In a preferred embodiment of the present invention a background memory manager (BMM) for managing a memory in a data processing system is provided, the BMM comprising circuitry for transferring data to and from an outside device and to and from a memory, a memory state map associated with the memory, and a communication link to a processor. The BMM manages the memory, performing all data transfers between the

4

outside device and the memory, and maintains the memory state map according to memory transactions made.

In preferred embodiments the BMM, after storing a data structure into the memory, provides a data identifier for the structure on the link to the processor. Also in preferred embodiments, the BMM, in making memory transactions, updates the memory state map to the new memory state, keeping track of regions occupied by valid data and regions not occupied by valid data.

In some embodiments the BMM, in response to a signal on the processor link that the processor is finished with certain identified data in the memory, copies the data from the memory to another device, and updates the memory state map to indicate the region of the data copied. There may further be an interrupt handler allowing a remote data source to interrupt the BMM when data is available to be transferred to the memory. The BMM in preferred embodiments is particularly suited for handling data packets in a packet processing router.

In another aspect of the invention a data processing system is provided, comprising a processor, a memory coupled to the processor, and a background memory manager coupled to the memory and the processor, the background memory manager including circuitry for transferring data to and from an outside device and to and from the memory, and a memory state map associated with the memory. The BMM manages the memory, performing all data transfers between the outside device and the memory, and maintains the memory state map according to memory transactions made.

In preferred embodiments of the system the BMM, after storing a data structure into the memory, provides a data identifier for the structure to the processor. Also in preferred embodiments the BMM, in making memory

5

transactions, updates the memory state map to the new memory state, keeping track of regions occupied by valid data and regions not occupied by valid data.

In some embodiments of the system the BMM, in response to a signal from the processor that the processor is finished with certain identified data in the memory, copies the data from the memory to another device, and updates the memory state map to indicate the region of the data copied. In some embodiments there is an interrupt handler allowing a remote data source to interrupt the BMM when data is available to be transferred to the memory. The data processing system is particularly suited to processing data packets in Internet packet processors.

In yet another aspect of the invention a network packet router is provided, comprising an input/output (I/O) device for receiving and sending packets on the network, a processor, a memory coupled to the processor, and a background memory manager coupled to the memory and the processor, the background memory manager including circuitry for transferring packets to and from the I/O device and to and from the memory, and a memory state map associated with the memory. The BMM manages the memory, performing all packet transfers between the I/O device and the memory, and maintains the memory state map according to memory transactions made.

In a preferred embodiment the BMM, after storing a packet into the memory, provides a data identifier for the packet to the processor. Also in a preferred embodiment the BMM, in making memory transactions, updates the memory state map to the new memory state, keeping track of regions occupied by valid packets and regions not occupied by valid packets.

In some embodiments the BMM, in response to a signal that the processor is finished with a packet in the memory, copies the packet from the memory to the I/O device, and updates the memory state map to indicate the region of the packet copied. There may also be an interrupt handler allowing the I/O device to interrupt the BMM when packets are available to be transferred to the memory.

In still another aspect of the present invention a method for managing a memory in a data processing system is provided, comprising the steps of (a) transferring data to and from an outside device and to and from the memory by circuitry in a background memory manager (BMM); (b) updating a memory state map associated with the memory in the BMM each time a memory transaction is made; and (c) notifying a processor with memory state data each time a change is made.

In preferred embodiments of the method, in step (c), the BMM, after storing a data structure into the memory, provides a data identifier for the structure on the link to the processor. Also in preferred embodiments the BMM, in step (b), in making memory transactions, updates the memory state map to the new memory state, keeping track of regions occupied by valid data and regions not occupied by valid data.

In some embodiments, in step (a), the BMM, in response to a signal that the processor is finished with certain identified data in the memory, copies the data from the memory to another device, and updates the memory state map to indicate the region of the data copied. There may further be a step for interrupting the BMM by the outside device when data is available to be transferred to the memory. The method is particularly well suited for processing data packets in a packet router, such as in the Internet.

In embodiments of the invention, taught in enabling detail below, for the first time an apparatus and methods are provided for complete background memory management, freeing processor power in systems like

Internet packet routers, to accomplish more processing, by not being required to do memory management.

Brief Description of the Drawing Figures

Fig. 1 is a simplified diagram of memory management by direct I/O processing in the prior art.

Fig. 2 is a simplified diagram of memory management by direct memory access in the prior art.

Fig. 3 is a diagram of memory management by a Background Memory Manager in a preferred embodiment of the present invention.

Description of the Preferred Embodiments

Fig. 1 is a simplified diagram of memory management in a system 104 comprising a processor 100 and a memory 102 in communication with a device 106. In this example it is necessary to bring data from device 106 into memory 102 for processing, and sometimes to transmit processed data from memory 102 to device 106, if necessary. Management in this prior art example is by processor 100, which sends I/O commands to and receives responses and/or interrupts from device 106 via path 108 to manage movement of data between device 106 and memory 102 by path 110. The processor has to determine whether a data structure can fit into available space in memory, and has to decide where in the memory to store incoming data structures. Processor 100 has to fully map and track memory blocks

8

into and out of memory 102, and retrieves data for processing and stores results, when necessary, back to memory 102 via path 114. This memory management by I/O commands is very slow and cumbersome and uses processor resources quite liberally.

Fig. 2 is a simplified diagram of a processor system 200 in the prior art comprising a processor 100, a memory 102 and a direct memory access (DMA) device 202. This is the second of two systems by which data, in the conventional art, is brought into a system, processed, and sent out again, the first of which is by I/O operations as described just above. System 200 comprises a DMA device 202 which has built-in intelligence, which may be programmed by processor 100, for managing data transfers to and from memory 102. DMA device 202 is capable of compatible communication with external device 106, and of moving blocks of data between device 102 and 106, bi-directionally. The actual data transfers are handled by DMA device 202 transparently to processor 100, but processor 100 must still perform the memory mapping tasks, to know which regions of memory are occupied with data that must not be corrupted, and which regions are free to be occupied (overwritten) by new data.

In the system of Fig. 2 DMA processor 100 programs DMA device 202. This control communication takes place over path 204. DMA device 202 retrieves and transmits data to and from device 106 by path 208, and handles data transfers between memory 102 and processor 100 over paths 204 and 206.

In these descriptions of prior art the skilled artisan will recognize that paths 204, 206 and 208 are virtual representations, and that actual data transmission may be by various physical means known in the art, such as by parallel and serial bus structures operated by bus managers and the like, the bus structures interconnecting the elements and devices shown.

Fig. 3 is a schematic diagram of a system 300 including a Background Memory Manager (BMM) 302 according to an embodiment of the present invention. BMM 302 a hardware mechanism enabled to manage the memory in the background, i.e. with no intervention of the processor to decide where the data structure will be stored in the memory. Thus, the processor can utilize its resources for tasks other than to manage the memory.

The present invention in several embodiments is applicable in a general way to many computing process and apparatus. For example, in a preferred embodiment the invention is applicable and advantageous in the processing of data packets at network nodes, such as in routers in packet routers in the Internet. The packet processing example is used below as a specific example of practice of the present invention to specifically describe apparatus, connectivity and functionality.

In the embodiment of a packet router, device 106 represents input/output apparatus and temporary storage of packets received from and transmitted on a network over path 308. The network in one preferred embodiment is the well-known Internet network. Packets received from the Internet in this example are retrieved from device 106 by BMM 302, which also determines whether packets can fit into available regions in memory and exactly where to store each packet, and stores the packets in memory 102, where they are available to processor 100 for processing. Processor 100 places results of processing back in memory 102, where the processed packets are retrieved, if necessary, by BMM on path 312 and sent back out through device 106.

In the embodiment of Fig. 3 BMM 302 comprises a DMA 202 and also a memory state map 304. BMM 302 also comprises an interrupt handler in a preferred embodiment, and device 106 interrupts BMM 302 when a packet is received. When a packet is received, using DMA 202 and state map 304, the BMM performs the following tasks:

1. Decides whether a data structure fits into the memory. Whether the structure fits into memory, then, is a function of the size of the data packet and the present state of map 304, which indicates those regions of memory 102 that are available for new data to be stored.

2. If the incoming packet in step 1 above fits into memory, the BMM determines an optimal storage position. It was described above that there are advantages in sequential storage. Because of this, the BMM in a preferred embodiment stores packets into memory 102 in a manner to create a small number of large available regions, rather than a larger number of smaller available regions.

3. BMM 302 notifies processor 100 on path 310 when enough of the packet is stored, so that the processor can begin to perform the desired processing. An identifier for this structure is created and provided to the processor. The identifier communicates at a minimum the starting address of the packet in memory, and in some cases includes additional information.

4. BMM updates map 304 for all changes in the topology of the memory. This updating can be done in any of several ways, such as periodically, or every time a unit in memory is changed.

5. When processing is complete on a packet the BMM has stored in memory 102, the processor notifies BMM 302, which then transfers the processed data back to device 106. This is for the particular example of a

packet processing task. In some other embodiments data may be read out of memory 102 by MM 302 and sent to different devices, or even discarded. In notifying the BMM of processed data, the processor used the data structure identifier previously sent by the BMM upon storage of the data in memory 102.

6. The BMM updates map 304 again, and every time it causes a change in the state of memory 102. Specifically the BMM de-allocates the region or regions of memory previously allocated to the data structure and sets them as available for storage of other data structures, in this case packets.

It will be apparent to the skilled artisan that there may be many alterations in the embodiments described above without departing from the spirit and scope of the present invention. For example, a specific case of operations in a data packet router was illustrated. This is a single instance ~~/~~of a system wherein the invention may provide significant advantages. There are many other systems and processes that will benefit as well. Further, there are a number of ways BMM 302 may be implemented to perform the functionality described above, and there are many systems incorporating many different kinds of processors that might benefit. The present inventors are particularly interested in a system wherein a dynamic multi-streaming processor performs the functions of processor 100. For these reasons the invention should be limited only by the scope of the claims as listed below.

11